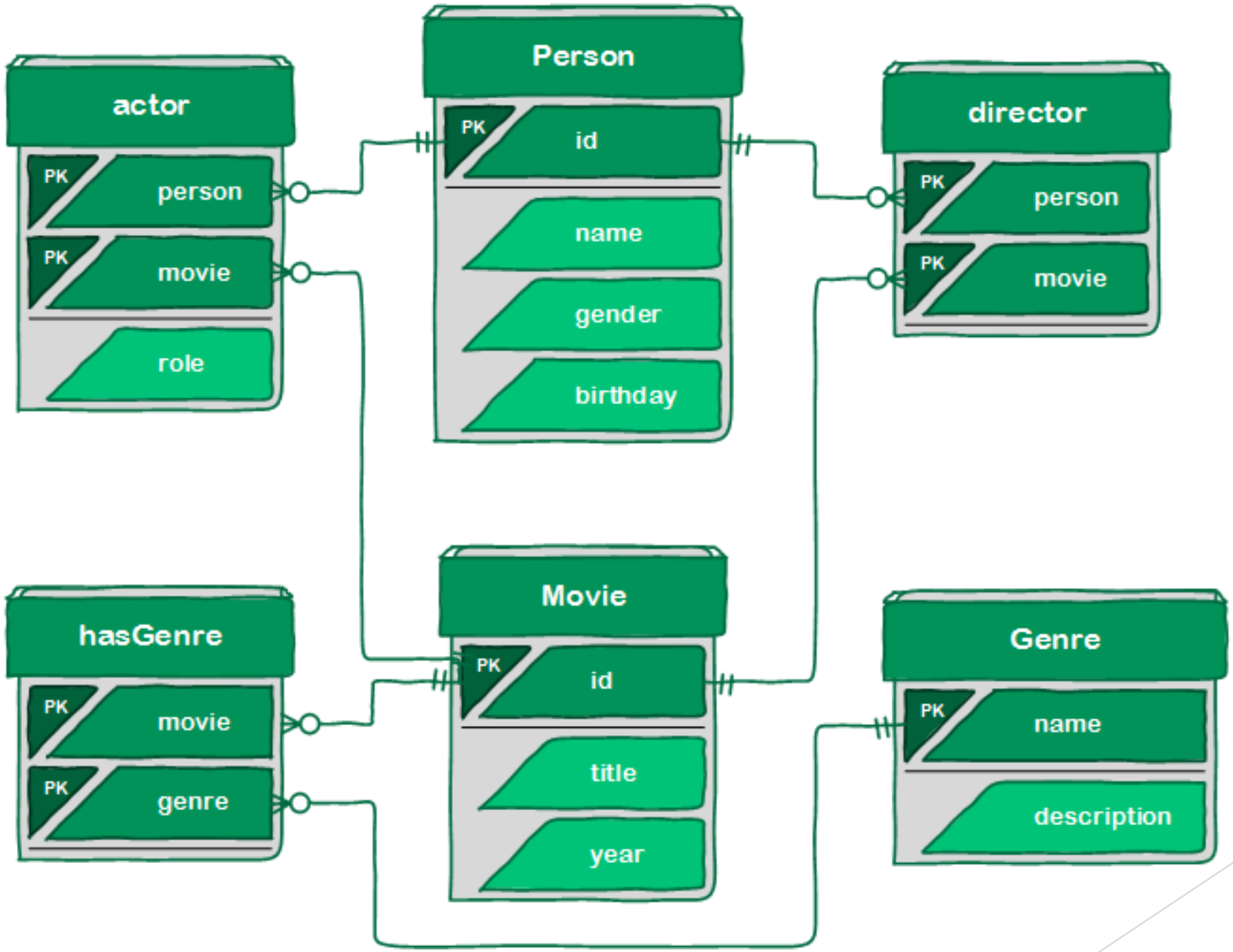
Mid-Term Solution

Database

2015/2016

Consider The following Scheme



Find the Title of all movies that have been created before 1970.

- **Relational Algebra**
- $R1 = \sigma_{year<1970}(Movie)$
- $R = \pi_{title}(R1)$
- **Relational Calculus**
- $R = \{title: t \mid Movie(id: i, title: t, year: y) \wedge y < 1970\}$
- **SQL**
- `SELECT title FROM Movie WHERE year<1970;`

Find the Name of all persons who participated in an “action” movie

- **Relational Algebra**
- $R1 = \pi_{id}\left(\sigma_{genre='action'}(Movie \bowtie_{id=movie} hasGenre)\right)$
- $R2 = \pi_{person}(R1 \bowtie_{id=movie} actor)$
- $R \ = \pi_{name}\left(R2 \bowtie_{person=id} Person\right)$
- **Relational Calculus**
- $R = \{name: n \mid Movie(id: m, title: t, year: y)$
  $\wedge\, hasGenre(movie: m, genre: g)$
  $\wedge\, actor(person: p, movie: m, role: r)$
  $\wedge\, Person(id: p, name: n, gender: gn, birthday: b)$
  $\wedge\, g = 'action'\}$

Find the Name of all persons who participated in an “action” movie

- **SQL**
- ```
  SELECT p.name FROM Person p join (actor a join
  (movie m join hasGenre h on m.id=h.movie) on a.movie = m.id)
  on p.id = a.person
  WHERE h.gerne = 'action';
  ```
  ```

Find Name of All persons who only played in the movie “The mighty Oracle” and did not play in any other movie.

- **Relational Algebra**
- $R1 = \pi_{person}(\sigma_{title \neq 'The\ mighty\ Oracle'}(Movie \bowtie_{id=movie} actor)$
- $R2 = \pi_{person}(actor)$
- $R3 = R2 - R1$
- $R = \pi_{name}(R3 \bowtie_{person=id} Person)$
- **Relational Calculus**
- $R = \{name: n \mid \forall_t (Movie(id: m, title: t, year: y) \wedge actor(person: p, movie: m, role: r) \wedge Person(id: p, name: n, gender: gn, birthday: b) \wedge t = 'The\ mighty\ Oracle')\}$

Find Name of All persons who only played in the movie “The mighty Oracle” and did not play in any other movie.

- **SQL**

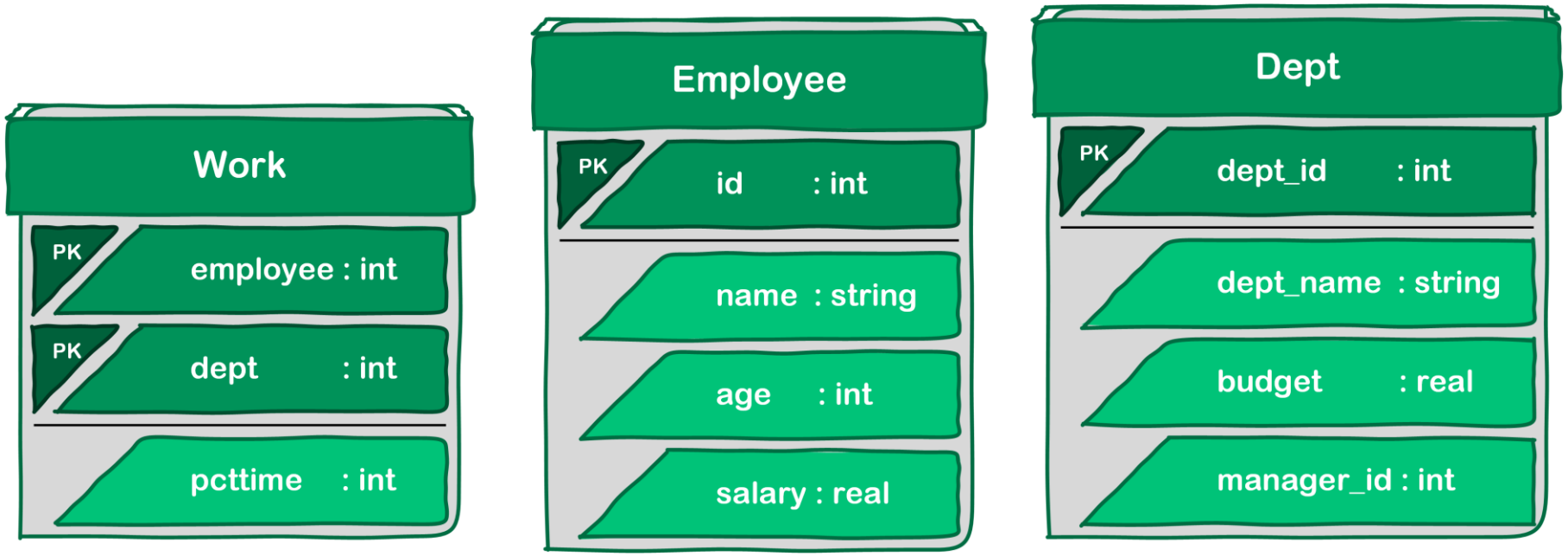
```
SELECT name FROM Person
WHERE id NOT IN(
     SELECT person FROM actor join Movie on movie=id
     WHERE title != 'The mighty Oracle'
)
AND id IN(
     SELECT person FROM actor join Movie on movie=id
     WHERE title = 'The mighty Oracle'
);
```

Find Name of All persons who only played in the movie “The mighty Oracle” and did not play in any other movie.

- **SQL – Other Solution**

```
SELECT name FROM Person
WHERE id IN(
     SELECT person FROM actor
     WHERE person NOT IN(
         SELECT person FROM actor join Movie on movie=id
         WHERE title != 'The mighty Oracle'
     )
)
```

Consider The following Scheme



Give an example for foreign key involves relation Dept.

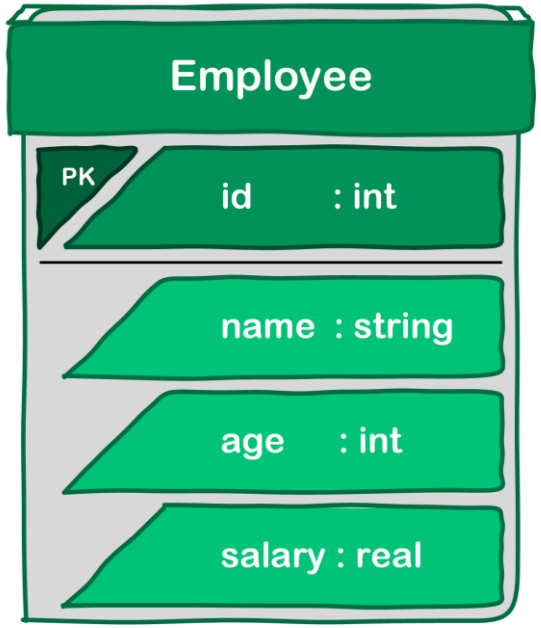
- **Relationship Between dept in work relation and dept_id in Dept relation.**

What are options for enforcing FK constraint when user delete a Dept tuple?

- **Cascade**
- **SET NULL**
- **SET DEFAULT**
- **NO Action**

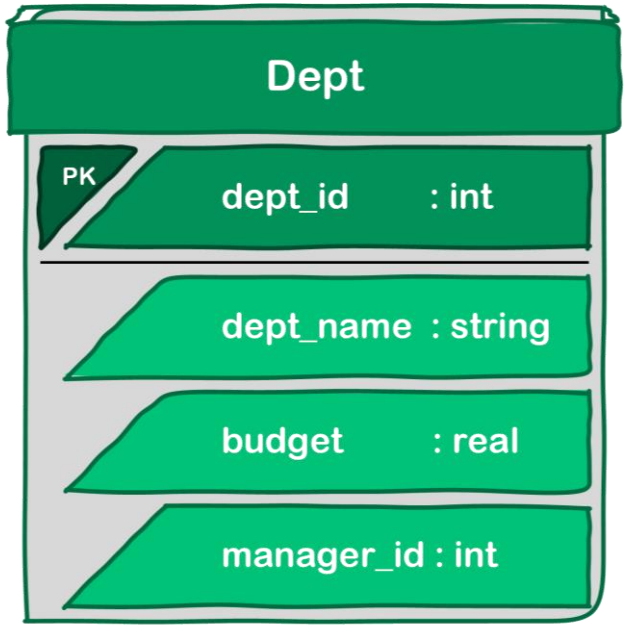
Create the scheme using SQL

```
CREATE TABLE Employee(
 id INT PRIMARY KEY,
 name VARCHAR(50),
 age INT,
 salary REAL
);
```



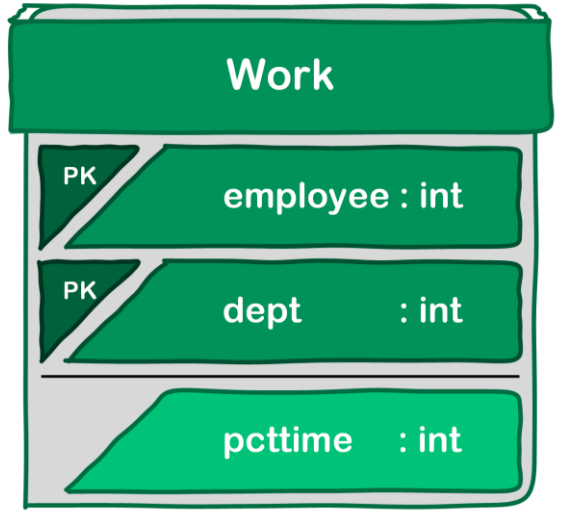
Create the scheme using SQL

```
CREATE TABLE Dept(
 dept_id INT PRIMARY KEY,
 dept_name VARCHAR(50),
 budget REAL,
 manager_id INT FOREIGN KEY
                  REFERENCES Employee(id)
);
```



Create the scheme using SQL

```
CREATE TABLE Work(
 employee INT FOREIGN KEY
              REFERENCES Employee(id),
 dept INT FOREIGN KEY
           REFERENCES Dept(dept_id),
 pcttime INT,
 CONSTRAINT Pk PRIMARY KEY(employee,dept)
);
```



Define the Dept relation in SQL so that every department must have a manager

- **We can change the definition…**
- ```
  ALTER TABLE Dept
  ALTER COLUMN manager_id INT FOREIGN KEY REFERENCES
  Employee(id) NOT NULL
  ```
- **We can use Check constraint…**
- ```
  ALTER TABLE Dept
  ADD CONSTRAINT ck_null CHECH(manager_id IS NOT NULL)
  ```
- **We also can do no thing because we have already used FOREIGN KEY constraint which refuse null values.**

Write SQL to add John Deo as an employee with id = 101, age = 32 and salary = 15,000

```
INSERT INTO Employee
(id, name, age, salary)
VALUES
(101, 'John Deo', 32, 15000);
```

Write SQL to give every employee a 10 percent raise.

```
UPDATE employee
SET salary = salary + 0.1 * salary;
```

Write SQL to delete the Toy deprtment, then explain what happens when this statement is executed.

- `DELETE FROM Dept WHERE dept_name = 'Toy';`
- **This statement will delete each department with name ‘Toy’ if there are no related data in an other table refere to it, that is because we use no action when we define our FOREIGN KEY constraint.**
- **No action means don’t execute the statement if it effects on other data in related relation.**